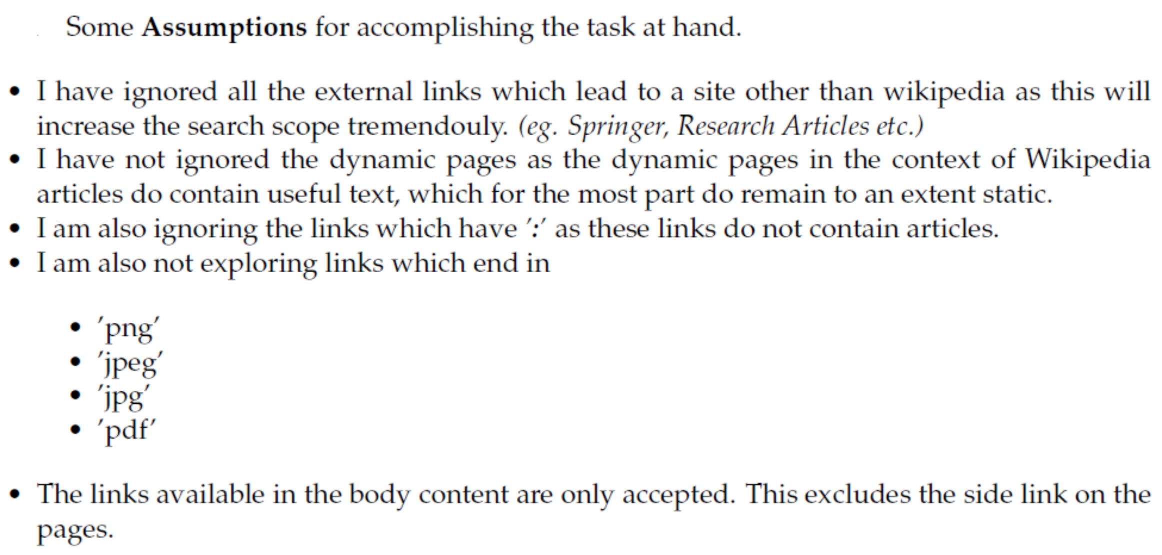
Experiment 3

Basic Web Crawling

Name:	**:**	**Abhishek N N**	**Register No**	**:**	**20BCE1025**
Faculty	:	Dr. Alok Chauhan	Slot	:	L51-L52 AB1-605B
Course	:	Web Mining Lab	Code	:	CSE3024
Programme	:	B.Tech CSE Core	Semester	:	Win – 22 - 23

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)


Some **Assumptions** for accomplishing the task at hand.

- I have ignored all the external links which lead to a site other than wikipedia as this will increase the search scope tremendouly. *(eg. Springer, Research Articles etc.)*
- I have not ignored the dynamic pages as the dynamic pages in the context of Wikipedia articles do contain useful text, which for the most part do remain to an extent static.
- I am also ignoring the links which have ':' as these links do not contain articles.
- I am also not exploring links which end in
  - 'png'
  - 'jpeg'
  - 'jpg'
  - 'pdf'
- The links available in the body content are only accepted. This excludes the side link on the pages.

Ex.1

Given a root URL, e.g., "https://en.wikipedia.org/wiki/Web_mining", Design a simple crawler to return all pages that contains a string “crawler" from this site.

The keyword **crawler** was hard to find because https://en.wikipedia.org/wiki/Web_mining automatically redirects to https://en.wikipedia.org/wiki/Data_mining

So I used **mining** instead

Code:

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
import re

pages = set()


pageUrl="https://en.wikipedia.org/wiki/Web_mining"
html = urlopen(pageUrl.format(pageUrl))
bs = BeautifulSoup(html, 'html.parser')
s=str(bs).lower()
i=s.find('mining')
if i!=-1:
    print(pageUrl)
    print(s[max(0,i-100):min(len(s)-1,i+100)])
    print('-'*20)
```

```
for link in bs.find_all('a', href=re.compile('^(/wiki/)')):
    if 'href' in link.attrs:
        if link.attrs['href'] not in pages:
            #We have encountered a new page
            newPage = link.attrs['href']
            newPage="https://en.wikipedia.org/"+newPage
            if newPage in pages:
                continue
            pages.add(newPage)
            html = urlopen(newPage.format(newPage))
            bs = BeautifulSoup(html, 'html.parser')
            s=str(bs).lower()
            i=s.find('mining')
            if i!=-1:
                print(newPage)
                print(s[max(0,i-100):min(len(s)-1,i+100)])
                print('-'*20)
```

output:

```
Output exceeds the size limit. Open the full output data in a text editor
https://en.wikipedia.org/wiki/Web_mining
pe html>

<html class="client-nojs" dir="ltr" lang="en">
<head>
<meta charset="utf-8"/>
<title>data mining - wikipedia</title>
<script>document.documentelement.classname="client-js";rlconf={"wgbreakfr
--------------------
https://en.wikipedia.org//wiki/Special:WhatLinksHere/Data_mining
s="client-nojs" dir="ltr" lang="en">
<head>
<meta charset="utf-8"/>
<title>pages that link to "data mining" - wikipedia</title>
<script>document.documentelement.classname="client-js";rlconf={"wgbreakf
--------------------
https://en.wikipedia.org//wiki/Special:RecentChangesLinked/Data_mining
gename":"recentchangeslinked","wgnamespacenumber":-1,"wgpagename":"special:r
mining","wgcurrevisionid":0,"wgrevisionid":0,"wgarticlei
--------------------
```

```
https://en.wikipedia.org//wiki/Data_mining
pe html>

<html class="client-nojs" dir="ltr" lang="en">
<head>
<meta charset="utf-8"/>
<title>data mining - wikipedia</title>
<script>document.documentelement.classname="client-js";rlconf={"wgbreakfr
--------------------
https://en.wikipedia.org//wiki/Talk:Data_mining
ml>

<html class="client-nojs" dir="ltr" lang="en">
<head>
<meta charset="utf-8"/>
<title>talk:data mining - wikipedia</title>
<script>document.documentelement.classname="client-js";rlconf={"wgbreakfr
--------------------
https://en.wikipedia.org//wiki/Cryptocurrency
mping-sublist">
</ul>
</li>
</ul>
</li>
<li class="vector-toc-list-item vector-toc-level-2" id="toc-mining">
<a class="vector-toc-link" href="#mining">
...
--------------------
https://en.wikipedia.org//wiki/Regression_analysis
="/wiki/machine_learning" title="machine learning">machine learning</a>
<br/>and <a href="/wiki/data_mining" title="data mining">data mining</a>
</th></tr><tr><td class="sidebar-image"><a class="image"
--------------------
```

Ex.2

Write a web crawler program which takes as input a url, a search word and maximum number of pages to be searched and returns as output all the web pages it searched till it found the search word on a web page or return failure.

Ex.3

Implement breadth-first-search and depth-first-search crawlers for Ex. 2.

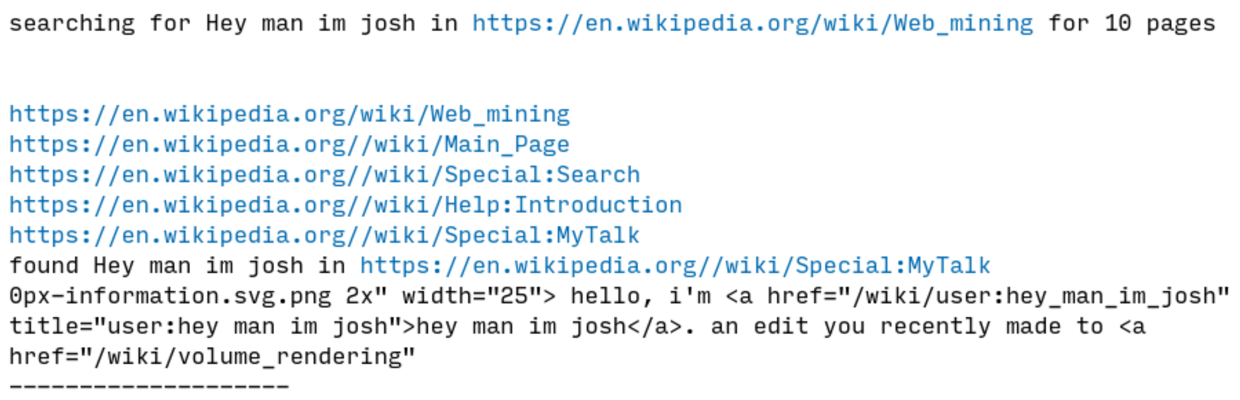
In Exp 2 either I have to implement breadth first search or depth first search so I will merge both both Ex into one

Depth First search

Code:

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
import re

def depthFirstSearch(url):
    global pages, searchWord, totalPages

    if url in pages:
        return
    pages.add(url)

    totalPages-=1

    print(url)
    html = urlopen(url.format(url))
    bs = BeautifulSoup(html, 'html.parser')
    s=str(bs).lower()
    i=s.find(searchWord.lower())
    if i!=-1:
        print("found",searchWord,"in",url)
        print(s[max(0,i-100):min(len(s)-1,i+100)])
        print('-'*20)
        totalPages=-1
        return
    for link in bs.find_all('a',
href=re.compile('^(/wiki/)')):
        if totalPages==-1:
            return
        if totalPages==0:
            print("searching terminated due to max pages 
reached")
            totalPages-=1
            return
        if 'href' in link.attrs:
            if link.attrs['href'] not in pages:
                newPage = link.attrs['href']
                newPage="https://en.wikipedia.org/"+newPage
                depthFirstSearch(newPage)

pages = set()
searchWord=input("Enter the word to search: ")
inputUrl=input("Enter the url to start with: ")
totalPages=int(input("Enter the number of pages to search: 
"))
print("searching 
for",searchWord,"in",inputUrl,"for",totalPages,"pages\n\n")
depthFirstSearch(inputUrl)
```

output:



```
searching for Hey man im josh in https://en.wikipedia.org/wiki/Web_mining for 10 pages


https://en.wikipedia.org/wiki/Web_mining
https://en.wikipedia.org//wiki/Main_Page
https://en.wikipedia.org//wiki/Special:Search
https://en.wikipedia.org//wiki/Help:Introduction
https://en.wikipedia.org//wiki/Special:MyTalk
found Hey man im josh in https://en.wikipedia.org//wiki/Special:MyTalk
0px-information.svg.png 2x" width="25"> hello, i'm <a href="/wiki/user:hey_man_im_josh"
title="user:hey man im josh">hey man im josh</a>. an edit you recently made to <a
href="/wiki/volume_rendering"
--------------------
```

but depth first search may get lost in deep web going out of context

```
searching for distributed ledger in https://en.wikipedia.org/wiki/Web_mining for 10
pages


https://en.wikipedia.org/wiki/Web_mining
https://en.wikipedia.org//wiki/Main_Page
https://en.wikipedia.org//wiki/Special:Search
https://en.wikipedia.org//wiki/Help:Introduction
https://en.wikipedia.org//wiki/Special:MyTalk
https://en.wikipedia.org//wiki/Special:MyContributions
https://en.wikipedia.org//wiki/Wikipedia:Contents
https://en.wikipedia.org//wiki/Portal:Current_events
https://en.wikipedia.org//wiki/Special:Random
https://en.wikipedia.org//wiki/Wikipedia:About
searching terminated due to max pages reached
```

Breadth First search

Code:

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
import re
from collections import deque

q = deque()
pages = set()

searchWord=input("Enter the word to search: ")
inputUrl=input("Enter the url to start with: ")
totalPages=int(input("Enter the number of pages to search:
"))
print("searching
for",searchWord,"in",inputUrl,"for",totalPages,"pages\n\n")

q.append(inputUrl)

while totalPages>0:
    totalPages-=1

    url = q.popleft()
    if url in pages:
        continue
    pages.add(url)

    html = urlopen(url.format(url))
    bs = BeautifulSoup(html, 'html.parser')
    s=str(bs).lower()
    i=s.find(searchWord.lower())
    if i!=-1:
        print("found",searchWord,"in",url)
        print(s[max(0,i-100):min(len(s)-1,i+100)])
        print('-'*20)
        break

    for link in bs.find_all('a',
href=re.compile('^(/wiki/)')):
        if 'href' in link.attrs:
            if link.attrs['href'] not in pages:
                newPage = link.attrs['href']
                newPage="https://en.wikipedia.org/"+newPage
                q.append(newPage)

if totalPages==0:
        print("searching terminated due to max pages
reached")
```

output:

```
searching for distributed ledger in https://en.wikipedia.org/wiki/Web_mining for 30 pages


https://en.wikipedia.org/wiki/Web_mining
https://en.wikipedia.org//wiki/Main_Page
https://en.wikipedia.org//wiki/Special:Search
https://en.wikipedia.org//wiki/Help:Introduction
https://en.wikipedia.org//wiki/Special:MyTalk
https://en.wikipedia.org//wiki/Special:MyContributions
https://en.wikipedia.org//wiki/Wikipedia:Contents
https://en.wikipedia.org//wiki/Portal:Current_events
https://en.wikipedia.org//wiki/Special:Random
https://en.wikipedia.org//wiki/Wikipedia:About
https://en.wikipedia.org//wiki/Help:Contents
https://en.wikipedia.org//wiki/Wikipedia:Community_portal
https://en.wikipedia.org//wiki/Special:RecentChanges
https://en.wikipedia.org//wiki/Wikipedia:File_upload_wizard
https://en.wikipedia.org//wiki/Special:WhatLinksHere/Data_mining
https://en.wikipedia.org//wiki/Special:RecentChangesLinked/Data_mining
https://en.wikipedia.org//wiki/Wikipedia:File_Upload_Wizard
https://en.wikipedia.org//wiki/Special:SpecialPages
https://en.wikipedia.org//wiki/Data_mining
https://en.wikipedia.org//wiki/Talk:Data_mining
https://en.wikipedia.org//wiki/Cryptocurrency
found distributed ledger in https://en.wikipedia.org//wiki/Cryptocurrency
 decentralized control, each cryptocurrency works through <a href="/wiki/distributed_ledger"
title="distributed ledger">distributed ledger</a> technology, typically a <a
href="/wiki/blockchain" title=
--------------------
```